DETAILED ACTION

Response to Amendment

1. Claims 1, 9, 20 have been amended. Claims 17-19 have been canceled. Claims 24-26 have been added. Claims 1-16, 20-26 are pending.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-5, 8-13, 16, 20, 21, 22, 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kitada (US 20030037163), in view of Applicant admitted prior art, and further in view of McCullough (US 20020010866 A1).

Claim 1. Kitada discloses *a method for fragmenting an incoming packet for transmission using a gateway device* (p11 [0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required).

Kitada does not teach the combination of these limitations: *a first outgoing packet and a second outgoing packet, storing a payload of the incoming packet in a plurality of storage units beginning in a first storage unit; transmitting the first outgoing packet being formed according to a predetermined portion of the payload stored in the first storage unit; and after transmitting the first outgoing packet, transmitting the second outgoing packet being formed according to a remaining portion of the payload stored in the storage units.*

In the same field of endeavor, Applicant admitted prior art discloses *a first outgoing packet* (Fig. 2 element 206) *and a second outgoing packet* (Fig. 2 element 210), *storing a payload of the incoming packet in a plurality of storage units beginning in a first storage unit* (Fig. 3 Buffers 1-12); *transmitting the first outgoing packet being formed according to a predetermined portion of the payload stored in the first storage unit* (Fig. 3 up to the copy point; a first storage unit covering up to the copy point and a second storage unit covering the remaining bits); *and after transmitting the first outgoing packet, transmitting the second outgoing packet being formed according to a remaining portion of the payload stored in the storage units* (Fig. 2 element 210 Fragment 2 element 208).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement (e.g. in hardware) Kitada's IP data encapsulation in accordance with PPPoE protocol.

Kitada discloses *a method for fragmenting an incoming packet for transmission* (p11 [0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, that Kitada's disclosure that fragment processing is required means that the payload of the incoming packet is fragmented into portions for transmission. Given an option of two portions, it would have been obvious to a person having ordinary skill in the art that these portions are either equal or one is larger than the other one. Hence when transmitting these portions either the packets are of equal size or one packet is larger than the other packet.

However, Kitada does not explicitly disclose *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the remaining portion corresponds to a majority of the payload of the incoming packet*.

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data

between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with Kitada suggests *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the remaining portion corresponds to a majority of the payload of the incoming packet*.

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with Kitada, thus modifying Kitada to have a small transfer followed by a large transfer - in other words when a size of the incoming packet exceeds a maximum packet size for a network connection; the remaining portion corresponds to a majority of the payload of the incoming packet - to avoid network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.

Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute

processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small ATM packet and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

Claims 2, 21. Kitada further teaches *the first and second outgoing packets are Point-to-Point Protocol over Ethernet frames* ([0185] perform fragment processing in order to encapsulate in accordance with PPPoE; FIG. 2C).

Claims 3, 23. Kitada does not teach the combination of these limitations: *generating a first outgoing sub-header according to a header of the incoming packet and the predetermined portion of the payload stored in the first storage unit; generating a second outgoing sub-header according to the header of the incoming packet or the first outgoing sub-header, and the remaining portion of the payload; including the first outgoing sub-header and the predetermined portion of the payload stored in the first storage unit in the first outgoing packet; and including the second outgoing sub-header and the remaining portion of the payload stored in the storage units in the second outgoing packet.*

In the same field of endeavor, Applicant admitted prior art discloses *generating a first outgoing sub-header according to a header of the incoming packet and the predetermined portion of the payload stored in the first storage unit* (Fig. 2 element 206, element 204– as modified as the result of the combination in the parent claim); *generating a second outgoing sub-header according to the header of the incoming packet or the first outgoing sub-header, and the remaining portion of the payload* (Fig. 2 element 210, element 208– as modified as the result of the combination in the parent claim); *including the first outgoing sub-header and the predetermined portion of the payload stored in the first storage unit in the first outgoing packet* (Fig. 3 up to the copy point– as modified as the result of the combination in the parent claim); *and including the second outgoing sub-header and the remaining portion of the payload stored in the storage units in the second outgoing packet* (Fig. 3 after the Copy point– as modified as the result of the combination in the parent claim).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claims 4, 22. Kitada further teaches *the incoming packet is an Internet Protocol packet and the header of the incoming packet is the IP header of the incoming IP packet* ([0185] perform fragment processing on IP packets).

Claim 5. Kitada does not teach the combination of these limitations: *the first outgoing sub-header is a first IP header corresponding to the predetermined portion of the payload stored in the first storage unit and the incoming IP header, and the second outgoing sub-header is a second IP header corresponding to the remaining portion of the payload, and the incoming IP header or the first outgoing sub-header.*

In the same field of endeavor, Applicant admitted prior art discloses *the first outgoing sub-header is a first IP header corresponding to the predetermined portion of the payload stored in the first storage unit* (Fig. 3 up to the copy point– as modified as the result of the combination in the parent claim) *and the incoming IP header* (Fig. 2 element 206, element 204), *and the second outgoing sub-header is a second IP header corresponding to the remaining portion of the payload* (Fig. 3 after the Copy point– as modified as the result of the combination in the parent claim), *and the incoming IP header or the first outgoing sub-header* (Fig. 2 element 210, element 208– as modified as the result of the combination in the parent claim).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claim 8. Kitada does not teach the combination of these limitations: *the first outgoing sub-header and the first fragment are included as a first outgoing payload of the first*

outgoing packet, and the second outgoing sub-header and the second fragment are included as a second outgoing payload of the second outgoing packet.

In the same field of endeavor, Applicant admitted prior art discloses *the first outgoing sub-header and the first fragment are included as a first outgoing payload of the first outgoing packet* (Fig. 2 element 206– as modified as the result of the combination in the parent claim), *and the second outgoing sub-header and the second fragment are included as a second outgoing payload of the second outgoing packet* (Fig. 2 element 210– as modified as the result of the combination in the parent claim).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claim 9. Kitada discloses *a method for fragmenting an incoming packet for inclusion and a gateway device* ([0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required)

Kitada does not teach the combination of these limitations: *a first outgoing packet and a second outgoing packet, storing a payload of the incoming packet as a first*

fragment and a second fragment in a plurality of storage units; including the first fragment in the first outgoing packet; and after including the first fragment in the first outgoing packet, including the second fragment in the second outgoing packet.

Applicant admitted prior art discloses *a first outgoing packet* (Fig. 2 element 206) *and a second outgoing packet* (Fig. 2 element 210), *storing a payload of the incoming packet as a first fragment and a second fragment in a plurality of storage units* (Fig. 3 Buffers 1-12)*; including the first fragment in the first outgoing packet, and after including the first fragment in the first outgoing packet, including the second fragment in the second outgoing packet* (Fig. 2 element 210 Fragment 2 element 208).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Kitada discloses *a method for fragmenting an incoming packet for transmission* (p11 [0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, that Kitada's disclosure that fragment processing is required means that the payload of the incoming packet is fragmented into portions for transmission. Given an option of two portions, it would have been obvious to a person having ordinary skill in the art that these portions are either equal or one is larger than the other one. Hence when transmitting these portions either the packets are equal size or one packet is larger than the other packet.

However, Kitada does not disclose that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the second fragment corresponds to a majority of the payload of the incoming packet.*

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with Kitada suggests that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the second fragment corresponds to a majority of the payload of the incoming packet.*

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with Kitada, thus modifying Kitada to have a small transfer followed by a large transfer, to avoid network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.

Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small ATM packet, and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

Claim 10. Kitada further teaches *the incoming packet is an Internet Protocol packet received in an Ethernet frame and the first and second outgoing packets are Point-to-Point Protocol over Ethernet frames* ([0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C).

Claim 11. Kitada does not teach the combination of these limitations: *generating a first outgoing sub-header and a second outgoing sub-header according to the first fragment, the second fragment, and a header of the incoming packet; including the first outgoing sub-header in the first outgoing packet; and including the second outgoing sub-header in the second outgoing packet*.

In the same field of endeavor, Applicant admitted prior art discloses *generating a first outgoing sub-header and a second outgoing sub-header according to the first fragment, the second fragment, and a header of the incoming packet; including the first outgoing sub-header in the first outgoing packet; and including the second outgoing sub-header in the second outgoing packet* (Fig. 2 elements 200, 206, 210).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claim 12. Kitada further teaches *the incoming packet is an incoming Internet Protocol packet and the header of the incoming packet is the IP header of the incoming IP packet* ([0185] perform fragment processing on IP packets).

Claim 13. Kitada does not teach the combination of these limitations: *the first outgoing sub-header is a first outgoing IP header generated corresponding to the first fragment and the IP header of the incoming IP packet, and the second outgoing sub-header is a second outgoing IP header generated corresponding to the second fragment, and the IP header of the incoming IP packet or the first outgoing sub-header.*

In the same field of endeavor, Applicant admitted prior art discloses *the first outgoing sub-header is a first outgoing IP header generated corresponding to the first fragment and the IP header of the incoming IP packet, and the second outgoing sub-header is a second outgoing IP header generated corresponding to the second fragment, and the IP header of the incoming IP packet or the first outgoing sub-header* (Fig. 2 elements 200, 206, 210).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claim 16. Kitada does not teach the combination of these limitations: *the first outgoing sub-header and the first fragment are included in a payload of the first outgoing packet,*

and the second outgoing sub-header and the second fragment are included in a payload of the second outgoing packet.

In the same field of endeavor, Applicant admitted prior art discloses *the first outgoing sub-header and the first fragment are included in a payload of the first outgoing packet* (Fig. 2 element 206– as modified as the result of the combination in the parent claim), *and the second outgoing sub-header and the second fragment are included in a payload of the second outgoing packet* (Fig. 2 element 210– as modified as the result of the combination in the parent claim).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Claim 20. Kitada discloses *a gateway device and a method for fragmenting an incoming packet for transmission as first and second outgoing packets* (p11 [0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required).

Kitada does not teach the combination of these limitations:

storing payload of the incoming packet in a storage unit;

transmitting the first outgoing packet being formed according to a predetermined portion of the payload stored in the storage unit; and

after transmitting the first outgoing packet, transmitting the second outgoing packet being formed according to a remaining portion of the payload stored in the storage unit.

In the same field of endeavor, Applicant admitted prior art discloses *storing payload of the incoming packet in a storage unit* (Fig. 3 Buffers 1-12)*; transmitting the first outgoing packet* (Fig. 2 element 206) *being formed according to a predetermined portion of the payload stored in the storage unit* (Fig. 3 up to the copy point; a first storage unit covering up to the copy point and a second storage unit covering the remaining bits)*; and after transmitting the first outgoing packet, transmitting the second outgoing packet* (Fig. 2 element 210) *being formed according to a remaining portion of the payload stored in the storage unit* (Fig. 2 element 210 Fragment 2 element 208).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the Applicant's admitted prior art with Kitada to implement Kitada's IP data encapsulation in accordance with PPPoE protocol.

Kitada discloses *a method for fragmenting an incoming packet for transmission* (p11 [0185] perform fragment processing on IP packets in order to encapsulate the IP packets in accordance with PPPoE; FIG. 2C, [0188] when the IP data is encapsulated in accordance with PPPoE, the overhead of 8 bytes is included in the payload portion of 1,500 bytes, Therefore, the maximum transfer unit of the IP packet is reduced to 1,492 bytes. Hence, since the IP packet is transmitted in accordance with PPPoE, fragment processing is required).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, that Kitada's disclosure that fragment processing is required means that the payload of the incoming packet is fragmented into portions for transmission. Given an option of two portions, it would have been obvious to a person having ordinary skill in the art that these portions are either equal or one is larger than the other one. Hence when transmitting these portions either the packets are of equal size or one packet is larger than the other packet.

However, Kitada does not disclose that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the size of the second outgoing packet is larger than that of the first outgoing packet*.

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data

between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with Kitada suggests *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the size of the second outgoing packet is larger than that of the first outgoing packet*.

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with Kitada, thus modifying Kitada to have a small transfer followed by a large transfer, to avoid network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.

Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size

packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small ATM packet and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

4. Claims 6-7, 14-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kitada, in view of Applicant admitted prior art, further in view of McCullough as applied to claims 1, 9 above, and further in view of Kitamura (US 20030065799).

Claim 6. Kitada does not teach the combination of these limitations: *generating the first outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming IP header according to the predetermined portion of the payload stored in the first storage unit*.

In the same field of endeavor, Kitamura discloses *generating the first outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming IP header according to the predetermined portion of the payload stored in the first storage unit* (Fig. 8, p6 [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with Kitada to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 7. Kitada does not teach the combination of these limitations: *generating the second outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming packet IP header or the first outgoing sub-header according to the remaining portion of the payload stored in the storage units.*

In the same field of endeavor, Kitamura discloses *generating the second outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming packet IP header or the first outgoing sub-header according to the remaining portion of the payload stored in the storage units* (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with Kitada to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 14. Kitada does not teach the combination of these limitations: *generating the first outgoing sub-header modifies the MF, Offset, Length, and checksum fields of the incoming packet IP header according to the first fragment.*

In the same field of endeavor, Kitamura discloses *generating the first outgoing sub-header modifies the MF, Offset, Length, and checksum fields of the incoming packet IP header according to the first fragment* (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with Kitada to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 15. Kitada does not teach the combination of these limitations: *generating the second outgoing sub-header modifies the MF, offset, length, and checksum fields of the incoming packet IP header or the first outgoing sub-header according to the second fragment.*

In the same field of endeavor, Kitamura discloses *generating the second outgoing sub-header modifies the MF, offset, length, and checksum fields of the incoming packet IP header or the first outgoing sub-header according to the second fragment* (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a

type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with Kitada to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

5. Claims 24-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kitada, in view of Applicant admitted prior art, further in view of McCullough as applied to claims 1, 9, 20 above, and further in view of Starkovich (US 6212546 B1).

Claims 24-26. The combination of references applied to the parent claim suggest all of the limitations of the parent claim and *the methods of claims 1, 9, 20, wherein the first outgoing packet is stored in a single storage unit of the computer memory of the gateway device and the second outgoing packet is stored in a plurality of storage units of the computer memory of the gateway device,*

Kitada does not explicitly teach *wherein as soon as all data stored in a respective storage unit is transmitted, the respective storage unit is freed for other uses by the gateway device*.

In the same field of endeavor, Starkovich discloses (12:18-26 if the memory required for buffer is greater than that of the input buffer, connector must reallocate the

memory using the function reallocate. It is important to reallocate the same buffer, so the gateway can free the memory, before exiting the Process Request function).

The combination of Starkovich with Kitada suggests *wherein as soon as all data stored in a respective storage unit is transmitted, the respective storage unit is freed for other uses by the gateway device*.

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Starkovich with Kitada, thus further modifying Kitada to free, for other uses by the gateway device, the respective storage unit as soon as all data stored in a respective storage unit is transmitted - so that the memory that is no longer in use is made available for other services by the memory management system.

6. Claims 1-5, 8-13, 16, 20, 21, 22, 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant admitted prior art (AAPA), in view of McCullough.

Claim 1. AAPA discloses *a method for fragmenting an incoming packet for transmission* as *a first outgoing packet using a gateway device* (Fig. 2 element 206) *and a second outgoing packet* (Fig. 2 element 210), *storing a payload of the incoming packet in a plurality of storage units beginning in a first storage unit* (Fig. 3 Buffers 1-12); *transmitting the first outgoing packet being formed according to a predetermined portion of the payload stored in the first storage unit* (Fig. 3 up to the copy point; a first storage unit covering up to the copy point and a second storage unit covering the remaining

bits); *and after transmitting the first outgoing packet, transmitting the second outgoing packet being formed according to a remaining portion of the payload stored in the storage units* (Fig. 2 element 210 Fragment 2 element 208).

AAPA does not disclose that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the remaining portion corresponds to a majority of the payload of the incoming packet.*

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with AAPA suggests *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the remaining portion corresponds to a majority of the payload of the incoming packet.*

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with AAPA, thus modifying AAPA to have a small transfer followed by a large transfer, to avoid

network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.

Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small ATM packet and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

Claims 2, 21. AAPA further teaches *the first and second outgoing packets are Point-to-Point Protocol over Ethernet frames* (FIG. 2).

Claims 3, 23. AAPA further discloses *generating a first outgoing sub-header according to a header of the incoming packet and the predetermined portion of the payload stored in the first storage unit* (Fig. 2 element 206, element 204– as modified as the result of the combination in the parent claim); *generating a second outgoing sub-header according to the header of the incoming packet or the first outgoing sub-header, and the remaining portion of the payload* (Fig. 2 element 210, element 208– as modified as the result of the combination in the parent claim); *including the first outgoing sub-header and the predetermined portion of the payload stored in the first storage unit in the first outgoing packet* (Fig. 3 up to the copy point – AAPA had been modified in the parent claims 1 and 20 so that the copy point now represents a small portion of the incoming packet. Fig. 3 is the replacement Fig. 3 Prior art which was filed on 07/31/2008); *and including the second outgoing sub-header and the remaining portion of the payload stored in the storage units in the second outgoing packet* (Fig. 3 after the Copy point– as modified as the result of the combination in the parent claim).

Claims 4, 22. AAPA further teaches *the incoming packet is an Internet Protocol packet and the header of the incoming packet is the IP header of the incoming IP packet* (Fig. 2).

Claim 5. AAPA further discloses *the first outgoing sub-header is a first IP header corresponding to the predetermined portion of the payload stored in the first storage unit* (Fig. 3 up to the copy point – as modified as the result of the combination in the parent

claim) *and the incoming IP header* (Fig. 2 element 206, element 204), *and the second outgoing sub-header is a second IP header corresponding to the remaining portion of the payload* (Fig. 3 after the Copy point– as modified as the result of the combination in the parent claim), *and the incoming IP header or the first outgoing sub-header* (Fig. 2 element 210, element 208).

Claim 8. AAPA further discloses *the first outgoing sub-header and the first fragment are included as a first outgoing payload of the first outgoing packet* (Fig. 2 element 206– as modified as the result of the combination in the parent claim), *and the second outgoing sub-header and the second fragment are included as a second outgoing payload of the second outgoing packet* (Fig. 2 element 210– as modified as the result of the combination in the parent claim).

Claim 9. AAPA discloses *a gateway device a method for fragmenting an incoming packet for inclusion in* a *first outgoing packet* (Fig. 2 element 206) *and a second outgoing packet* (Fig. 2 element 210), *storing a payload of the incoming packet as a first fragment and a second fragment in a plurality of storage units* (Fig. 3 Buffers 1-12); *including the first fragment in the first outgoing packet*, *and after including the first fragment in the first outgoing packet, including the second fragment in the second outgoing packet* (Fig. 2 element 210 Fragment 2 element 208).

AAPA does not disclose that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the second fragment corresponds to a majority of the payload of the incoming packet.*

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with AAPA suggests that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the second fragment corresponds to a majority of the payload of the incoming packet.*

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with AAPA, thus modifying AAPA to have a small transfer followed by a large transfer, to avoid network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another

motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.
Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small ATM packet, and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

Claim 10. AAPA further teaches *the incoming packet is an Internet Protocol packet received in an Ethernet frame and the first and second outgoing packets are Point-to-Point Protocol over Ethernet frames* (FIG. 2).

Claim 11. AAPA discloses *generating a first outgoing sub-header and a second outgoing sub-header according to the first fragment, the second fragment, and a header of the incoming packet; including the first outgoing sub-header in the first outgoing*

packet; and including the second outgoing sub-header in the second outgoing packet (Fig. 2 elements 200, 206, 210).

Claim 12. AAPA further teaches *the incoming packet is an incoming Internet Protocol packet and the header of the incoming packet is the IP header of the incoming IP packet* (FIG. 2).

Claim 13. AAPA discloses *the first outgoing sub-header is a first outgoing IP header generated corresponding to the first fragment and the IP header of the incoming IP packet, and the second outgoing sub-header is a second outgoing IP header generated corresponding to the second fragment, and the IP header of the incoming IP packet or the first outgoing sub-header* (Fig. 2 elements 200, 206, 210).

Claim 16. AAPA discloses *the first outgoing sub-header and the first fragment are included in a payload of the first outgoing packet* (Fig. 2 element 206– as modified as the result of the combination in the parent claim), *and the second outgoing sub-header and the second fragment are included in a payload of the second outgoing packet* (Fig. 2 element 210– as modified as the result of the combination in the parent claim).

Claim 20. AAPA discloses *a gateway device and a method for fragmenting an incoming packet for transmission as first and second outgoing packets, storing payload of the incoming packet in a storage unit* (Fig. 3 Buffers 1-12);

transmitting the first outgoing packet (Fig. 2 element 206) *being formed according to a predetermined portion of the payload stored in the storage unit* (Fig. 3 up to the copy point; a first storage unit covering up to the copy point and a second storage unit covering the remaining bits)*; and*
after transmitting the first outgoing packet, transmitting the second outgoing packet (Fig. 2 element 210) *being formed according to a remaining portion of the payload stored in the storage unit* (Fig. 2 element 210 Fragment 2 element 208).

AAPA does not disclose that *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the size of the second outgoing packet is larger than that of the first outgoing packet*.

In the same field of endeavor, McCullough discloses ([0078] The bundle manager fragments a packet by comparing the size of the packet with the transmission unit MTU, which for underlying PPP links is 1500 bytes, [0079] In the gateway device, the fragment size is set at configuration time to 50% of the PPP MTU, this setting can be overridden, the upper limit is the full MTU. when there is a large transfer of data between peers, the bundle manager distributes a 1500 byte fragment on each available link in a round-robin fashion, or numerous small transfers are interleaved with fewer large transfers. fragment size is tuned for different circumstances to achieve the best aggregate throughput).

The combination of McCullough with AAPA suggests *when a size of the incoming packet exceeds a maximum packet size for a network connection*; *the size of the second outgoing packet is larger than that of the first outgoing packet*.

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of the McCullough with AAPA, thus modifying AAPA to have a small transfer followed by a large transfer, to avoid network congestion and adaptively achieve the best aggregate throughput. E.g., when the network is congested – it is advantages to send a small size packet first, this will keep the connection alive and have the least impact on bandwidth resources. Another motivation is a fast start – by sending out a small size packet first, the receiver would receive some data faster and would start the communication earlier.

Elaborating on the scenario wherein the size of the incoming packet exceeds a maximum packet size for a network connection - the network connection has to break up the packet for further transmission. A person having ordinary skill in the art knows that breaking up a packet consumes CPU and memory resources - which constitute processing time, delay, and an increase in latency. For time critical cases, where latency is important, and additionally for keeping a connection alive - a packet would need to arrive at the destination within a certain time frame. By sending out a small size packet first, the probability that the time constraints are met - will be enhanced. A third example is an ATM system. ATM packets are relatively small and when a large packet arrives at the network connection - the first packet that will be transmitted will be a small

ATM packet and the remaining portion of the original packet, waiting to be transmitted, corresponds to a majority of the payload of the incoming packet.

7. Claims 6-7, 14-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant admitted prior art, further in view of McCullough as applied to claims 1, 9 above, and further in view of Kitamura.

Claim 6. AAPA does not teach the combination of these limitations: *generating the first outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming IP header according to the predetermined portion of the payload stored in the first storage unit*.

In the same field of endeavor, Kitamura discloses *generating the first outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming IP header according to the predetermined portion of the payload stored in the first storage unit* (Fig. 8, p6 [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with AAPA to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 7. AAPA does not teach the combination of these limitations: *generating the second outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming packet IP header or the first outgoing sub-header according to the remaining portion of the payload stored in the storage units.*

In the same field of endeavor, Kitamura discloses *generating the second outgoing sub-header comprises modifying the MF, Offset, Length, and Checksum fields of the incoming packet IP header or the first outgoing sub-header according to the remaining portion of the payload stored in the storage units* (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with AAPA to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 14. AAPA does not teach the combination of these limitations: *generating the first outgoing sub-header modifies the MF, Offset, Length, and checksum fields of the incoming packet IP header according to the first fragment*.

In the same field of endeavor, Kitamura discloses *generating the first outgoing sub-header modifies the MF, Offset, Length, and checksum fields of the incoming*

packet IP header according to the first fragment (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with AAPA to follow the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

Claim 15. AAPA does not teach the combination of these limitations: *generating the second outgoing sub-header modifies the MF, offset, length, and checksum fields of the incoming packet IP header or the first outgoing sub-header according to the second fragment.*

In the same field of endeavor, Kitamura discloses *generating the second outgoing sub-header modifies the MF, offset, length, and checksum fields of the incoming packet IP header or the first outgoing sub-header according to the second fragment* (Fig. 8, [0118] expanded PPPoE portion, an upper layer identifier, a version, a type, a TOS, a data length, an identifier, a flag, a fragment offset, a TTL, an upper layer identifier, a header checksum, a source IP address, a destination IP address are written).

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Kitamura with AAPA to follow

the same arrangement as is customary in packet communications when one packet is fragmented into multiple packets.

8. Claims 24-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant admitted prior art, in view of McCullough as applied to claims 1, 9, 20 above, and further in view of Starkovich.

Claims 24-26. The combination of references applied to the parent claim suggest all of the limitations of the parent claim and *the methods of claims 1, 9, 20, wherein the first outgoing packet is stored in a single storage unit of the computer memory of the gateway device and the second outgoing packet is stored in a plurality of storage units of the computer memory of the gateway device,*

AAPA does not explicitly teach *wherein as soon as all data stored in a respective storage unit is transmitted, the respective storage unit is freed for other uses by the gateway device.*

In the same field of endeavor, Starkovich discloses (12:18-26 if the memory required for buffer is greater than that of the input buffer, connector must reallocate the memory using the function reallocate. It is important to reallocate the same buffer, so the gateway can free the memory, before exiting the Process Request function).

The combination of Starkovich with AAPA suggests *wherein as soon as all data stored in a respective storage unit is transmitted, the respective storage unit is freed for other uses by the gateway device*.

It would have been obvious to a person having ordinary skill in the art, at the time that the invention was made, to combine the teachings of Starkovich with AAPA, thus further modifying AAPA to free, for other uses by the gateway device, the respective storage unit as soon as all data stored in a respective storage unit is transmitted - so that the memory that is no longer in use is made available for other services by the memory management system.

Allowable Subject Matter

9. The following claims drafted by the examiner and considered to distinguish patentably over the art of record in this application, are presented to applicant for consideration:

a) Claim 3 written in independent form (base claim 1) and the limitations of claim 2 and 24 are incorporated into the resulting independent claim.

b) Claim 11 written in independent form (base claim 9) and the limitations of claim 10 and 25 are incorporated into the resulting independent claim.

c) Claim 23 written in independent form (base claim 20) and the limitations of claim 21 and 26 are incorporated into the resulting independent claim.

Response to Arguments

10. Applicant's arguments with respect to amended and new claims have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

11. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Hooman Houshmand whose telephone number is (571)270-1817. The examiner can normally be reached on Monday - Friday 8am - 5pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jay Patel can be reached on (571) 272-2988. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/H. H./
Examiner, Art Unit 2465

/Jayanti K. Patel/
Supervisory Patent Examiner, Art Unit 2465